Implicit Hierarchical Modeling for Cross-System Legal Case Retrieval

Zhengjia Zhong, Hongyi Lan, Yuhong Chen, Shengjia Hua, Kaiyan Liang
Xiaodong Li, Xianming Lin, Hui Li*, Rongrong Ji
{zhongplusplus, lanhongyi, yuhongchen, huashengjia, kaiyanliang}@stu.xmu.edu.cn,
{xdli, linxm, hui, rrji}@xmu.edu.cn,
Key Laboratory of Multimedia Trusted Perception and Efficient Computing,
Ministry of Education of China, Xiamen University, China

Abstract

Legal Case Retrieval (LCR) identifies relevant precedent cases to support legal reasoning and sentencing. A core challenge for cross-system LCR is the structural divergence between legal systems: civil-law systems provide explicit statutory hierarchies but lack inter-case citations, while common-law systems offer rich citation networks but lack uniform codified semantic hierarchies. Existing methods exploit only one side of this divide, remaining system-specific. We propose DSLaw, which unifies both systems by implicitly modeling hierarchical semantics through residual quantization and using the resulting discrete representations as semantic anchors for cross-case graph reasoning. By progressively decomposing legal semantics from coarse concepts to fine-grained distinctions, DSLaw constructs implicit connections even without explicit citations. Experiments on COLIEE2025 (common law) and ELAM (civil law) show DSLaw consistently outperforms strong baselines, while joint training further validates cross-system generalization. The implementation is provided at https://github.com/KDEGroup/DSLaw.

1 Introduction

Legal Case Retrieval (LCR) is a foundational task in legal AI. As shown in Fig. 1, LCR aims to identify precedent cases that are sufficiently similar to the current case (Feng et al., 2024). The applicability of a precedent depends on factors spanning from the legal domain to specific factual nuances, such as whether the case falls under criminal law, whether the charge is theft, and whether the theft was committed via cyber intrusion or physical break-in. This coarse-to-fine reasoning pattern is inherent to how legal decisions are made. Regardless of the legal system, judges reason from broad legal concepts toward fine-grained factual distinctions (Yu et al., 2022; Shao et al., 2023).

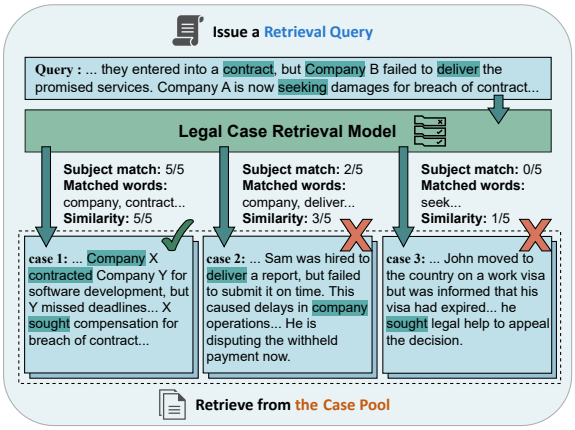

Figure 1: In LCR, queries are matched against a pool of cases. Various methods (e.g., semantic similarity) can be used to determine relevance and assign scores.

Yet legal systems organize such reasoning structures in fundamentally different ways (Feng et al., 2024). Civil-law systems provide explicit statutory hierarchies, with comprehensive, unified codes that classify legal domains, charge categories, and factual subtypes in a clear coarse-to-fine structure (Qin et al., 2024). However, cases in civil-law jurisdictions are not linked by explicit citation networks. Common-law systems, conversely, offer rich inter-case citation networks where precedent cases are explicitly referenced and distinguished (Tang et al., 2024), but lack uniform codified semantic hierarchies. Statutory classifications vary across jurisdictions and states, and the primary organizational principle is precedent-based rather than code-based. The two systems are structurally complementary. Civil law provides unified hierarchies but lacks inter-case citations; common law provides citation networks but lacks uniform semantic hierarchies.

Existing LCR methods exploit only one side. Methods like GEAR (Qin et al., 2024) build static legal trees tailored to civil-law systems and cannot be applied to common-law data, while CaseLink (Tang et al., 2024) constructs case graphs

* Corresponding author.

via BM25 (Robertson and Zaragoza, 2009) similarity and charge labels that drop sharply on civil-law benchmarks. LLM-based dense retrieval methods (Chalkidis et al., 2020; Xiao et al., 2021, 2024; Li et al., 2025) can be applied across different legal systems, but struggle to separate coarse-grained and fine-grained semantics in long legal documents, limiting their effectiveness for nuanced retrieval.

To unify LCR process across both legal systems, we need a representation that captures hierarchical semantics and inter-case relational patterns without relying on system-specific explicit structures. Such unification offers two advantages: it allows both types of structural information to jointly enhance training, and enables combining data from different legal systems to train a single model that generalizes across jurisdictions.

We propose **DSLaw**, Discrete Semantic Modeling for Law, which achieves this by implicitly modeling a Latent Legal Hierarchy (LLH) through residual quantization (RQ) and leveraging the resulting discrete representations as semantic anchors for cross-case graph reasoning. *The key insight is that discrete codes produced by RQ naturally form layered semantic levels from coarse legal domains to fine-grained factual nuances, and cases sharing the same code at any level are implicitly connected at that semantic level*. For civil-law systems where explicit citations are absent, shared codes provide implicit inter-case connections. For common-law systems where uniform semantic hierarchies are absent, the layered codes provide implicit semantic levels. DSLaw thus covers the structural gap on both sides through a single mechanism.

The key contributions of DSLaw are:

- **Structural Complementarity Across Legal Systems.** We identify that civil law provides unified hierarchies but lacks inter-case citations, while the opposite holds for common law. This divergence prevents existing LCR methods from generalizing across jurisdictions.
- **Implicit Legal Hierarchy Construction.** We adapt RQ to construct the latent legal hierarchy, decomposing continuous representations into layered discrete semantics. We further introduce the Residual Amplification Unit (RAU) to stabilize training and improve performance.
- **Discrete IDs as Semantic Anchors.** We show that the discrete IDs produced by RQ serve as semantic anchors, enabling cross-case graph reasoning without external structure annotations. Shared IDs create implicit connections between cases, allowing the graph to self-organize from internal semantics rather than relying on jurisdiction-specific structures.
- **Theoretical Insight.** We provide theoretical analyses showing that our design can progressively separate hard negatives and encourages a coarse-to-fine semantic encoding.
- **Cross-System Unification.** DSLaw achieves state-of-the-art results on both COLIEE2025 (common law) and ELAM (civil law), and joint training on both datasets further validates effective cross-system unification.

2 Related Work

Legal Case Retrieval. Existing LCR methods can be categorized into three approaches: (1) Sparse methods like TF-IDF (Aizawa, 2003) and BM25 (Robertson and Zaragoza, 2009) rely on lexical overlap and cannot capture hierarchical semantic abstraction. (2) LLM-based dense methods (Chalkidis et al., 2020; Xiao et al., 2021; Lu et al., 2021; Li et al., 2025) encode cases into flat vectors, entangling coarse-grained and fine-grained semantics in a single embedding without level-wise separation. (3) Structure-aware methods such as GEAR (Qin et al., 2024), KELLER (Deng et al., 2024), and CaseLink (Tang et al., 2024) incorporate legal organization, but each relies on system-specific explicit structures: GEAR builds statutory trees available only in civil-law systems, while CaseLink constructs graphs via BM25 similarity and charge labels that degrade on civil-law data. *A shared limitation across all three categories is the inability to model hierarchical semantics without system-specific annotations, which is precisely what cross-system unification requires.*

Residual Quantization. Vector quantization (VQ) compresses high-dimensional vectors by mapping them to the nearest code in a learnable codebook (Buzo et al., 1980). RQ generalizes VQ by iteratively encoding the residual errors via multiple codebooks, allowing for a coarse-to-fine approximation (Chen et al., 2010). VQ-VAE has extended VQ via discrete latent modeling (van den Oord et al., 2017), and RQ-VAE extends VQ-VAE as a multi-layer formulation (Zeghidour et al., 2022a), successfully applied in speech processing and recommender systems (Zeghidour et al., 2022b; Rajput et al., 2023). *This naturally motivates the use*

of RQ for hierarchical semantic encoding in LCR.

3 Preliminary

Vector quantization (VQ) (Buzo et al., 1980) is a prevalent vector compression paradigm that maps a continuous vector to a discrete ID with representational capacity and low reconstruction distortion by learning a codebook of centroids from the training vectors. In VQ, a codebook $\mathcal{E} \in \mathbb{R}^{K \times d}$ is a set of K learnable prototype vectors. Residual quantization (RQ) (Chen et al., 2010; Zeghidour et al., 2022b; Rajput et al., 2023) extends VQ by iteratively applying discrete ID mapping across L codebooks, enabling high-fidelity representations. Since discrete codes produced by RQ naturally form layered semantic levels, we use it as the basic representation module in DSLaw (Sec. 4.1). Hence, we first explain the background of RQ in the following.

Given an input vector e, let $r^{(0)} = e$ be the initial residual. At each layer $l \in \{0, 1, \ldots, L-1\}$, RQ selects the nearest codebook entry:

$$id^{(l)} = \arg\min_{k} \|r^{(l)} - \mathcal{E}^{(l)}[k]\|_2, \tag{1}$$

and obtain the quantized vector:

$$q^{(l)} = \mathcal{E}^{(l)}[id^{(l)}]. \tag{2}$$

The residual is then updated as:

$$r^{(l+1)} = r^{(l)} - q^{(l)}. \tag{3}$$

The output of RQ is a sequence of discrete semantic IDs $\{id^{(0)}, id^{(1)}, \ldots, id^{(L-1)}\}$, encoding the input in a coarse-to-fine manner where early stages capture dominant structure and later stages progressively refine residual details. RQ has been leveraged in neural audio codecs for signal representation (Zeghidour et al., 2022b) and in recommender systems for generating hierarchical item IDs (Rajput et al., 2023). LCR also follows a hierarchical reasoning process, progressing from broad legal categorization (e.g., criminal law vs. civil law) to fine-grained factual distinctions (e.g., theft via cyber intrusion vs. physical break-in). This structural alignment makes RQ a natural choice for modeling hierarchical representations in LCR.

4 Our Method DSLaw

Fig. 2 provides an overview of DSLaw, which consists of three components: (1) Legal Summary Extraction (LSE) for pre-processing raw legal documents into compact embeddings; (2) Hierarchical Semantics Encoding (HSE), which maps cases into discrete hierarchical representations via residual quantization; and (3) Case Representation Learning (CRL), which models inter-case relations through multi-level graph aggregation.

Legal documents are often lengthy and contain significant irrelevant content. LSE serves as a preprocessing step to convert each raw document into a compact embedding[1]. In summary, LSE extracts key sentences via TF-IDF and keyword matching, feeds them to an LLM for summarization, and encodes the resulting summary using a pretrained encoder followed by PCA-whitening to obtain legal document embedding $e_i \in \mathbb{R}^d$. Charge labels are processed similarly to obtain embeddings $e_y \in \mathbb{R}^d$.

4.1 Hierarchical Semantics Encoding (HSE)

Given the embedding of a legal document, HSE decomposes it into a layered discrete representation that captures hierarchical legal semantics, where shallow-layer IDs encode broad legal concepts and deeper-layer IDs progressively capture finer-grained factual distinctions. This layered structure forms the latent legal hierarchy. At each layer l, HSE selects a discrete ID $id_i^{(l)}$ from the corresponding codebook $\mathcal{E}^{(l)}$, yielding a sequence of semantic IDs for the case. The process starts from the initial residual, which is defined as:

$$r_i^{(0)} = En(e_i), \tag{4}$$

where $En()$ denotes a multi-layer MLP. The selection of nearest codebook entry and quantized vector follow Eq.(1)-(3) in Sec. 3.

However, in conventional RQ, the residual magnitude progressively diminishes across layers, leaving deep-layer residuals with insufficient discriminability to separate hard negative samples. To address this, we introduce the Residual Amplification Unit (RAU), which reshapes and amplifies the residual signal to preserve discriminative capacity in deeper layers. The residual for the next quantization layer is computed as:

$$r_i^{(l+1)} = \mathrm{RAU}(r_i^{(l)} - q_i^{(l)}), \tag{5}$$

where RAU is a lightweight residual modulation module defined as:

$$\mathrm{RAU}(x) = \mathrm{LayerNorm}(x + \mathrm{MLP}(x)), \tag{6}$$

with $\mathrm{MLP}(x) = W_2 \cdot \mathrm{swish}(W_1 \cdot x)$. By reshaping and amplifying the residual signal, RAU prevents deep-layer codebook collapse and encourages

[1]Details of LSE are provided in Appendix A

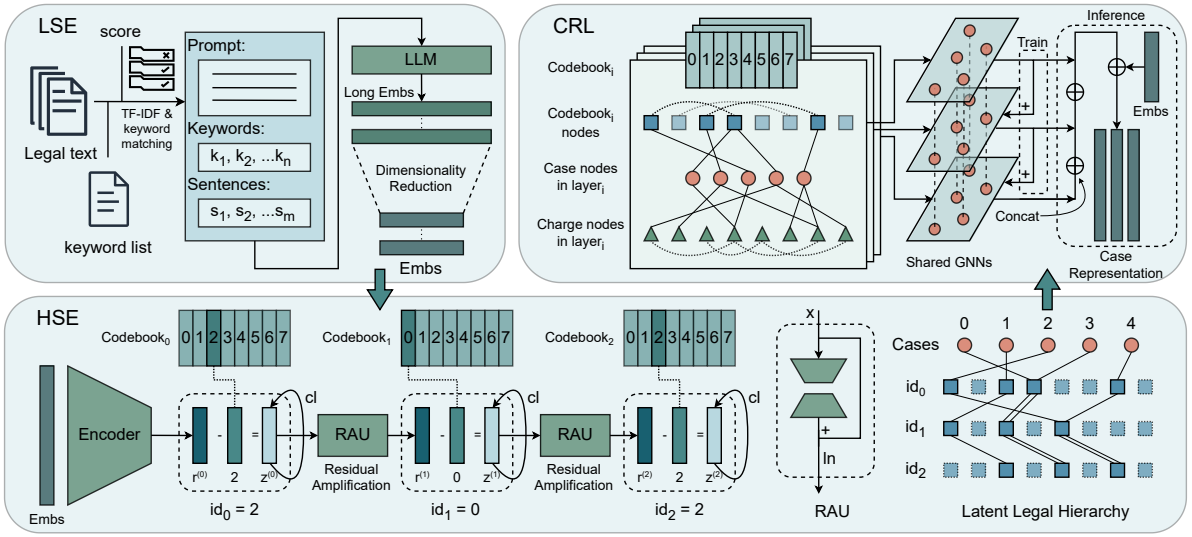

Figure 2: An overview of DSLaw. (1) **LSE** extracts meaningful legal spans from raw texts; (2) **HSE** supports the construction of a *latent legal hierarchy* by transforming input embeddings into multi-layer discrete codebook IDs; and (3) **CRL** performs hierarchical graph reasoning over the latent legal hierarchy to get the final case presentation.

sustained exploration of the latent space, yielding more diverse and expressive representations.

HSE outputs a sequence of discrete semantic IDs for each case, which serves as the structural foundation for the subsequent graph reasoning module. These IDs also provide interpretability[2]: Cases that share the first few discrete IDs mostly correspond to the same charge, and the last few discrete IDs for these same-charge cases capture finer-grained factual regularities.

4.2 Case Representation Learning (CRL)

Semantic IDs produced by HSE serve as semantic anchors for cross-case graph reasoning without requiring explicit structural signals. Cases sharing the same semantic ID at a given layer are implicitly aligned at the corresponding semantic level within the latent legal hierarchy, enabling latent semantic links beyond citation-based relations.

In both common-law and civil-law systems, these shared semantic concepts induce semantic relations between cases across multiple abstraction levels, allowing legal cases from different systems to be represented within a unified graph framework without relying on system-specific legal structures.

DSLaw bridges this structural gap by constructing hierarchical graphs over the discrete semantic IDs $\{id_i^{(0)}, \ldots, id_i^{(L-1)}\}$.

Graph Construction. For each quantization layer $l \in \{0, \ldots, L-1\}$, we build an undirected graph $\mathcal{G}^{(l)} = (\mathcal{V}^{(l)}, \mathcal{A}^{(l)})$, where the node set $\mathcal{V}^{(l)}$ contains case nodes $v_i^{(l)}$, charge nodes $v_y^{(l)}$, and code-ID nodes $v_{id}^{(l)}$ (from the l-th codebook).

The edges in $\mathcal{G}^{(l)}$ are designed to capture both explicit and latent structures:

- Case-Charge Edges: Connect case $v_i^{(l)}$ to its annotated charges.
- Case-ID Edges (The Semantic Bridge): Connect case $v_i^{(l)}$ to its assigned code-ID $v_{id_i}^{(l)}$. Crucially, if two cases connect to the same ID node, a 2-hop path is formed (Case A $\leftrightarrow$ ID $\leftrightarrow$ Case B), representing a *shared legal concept*. This allows message passing between structurally similar cases regardless of jurisdiction.
- Intra-Type Edges: Connect two code-ID nodes (or two charge nodes) if their embedding similarity ranks in the top-k neighbors.

At the l-th graph, case and charge nodes are initialized with the HSE residuals $r_i^{(l)}$ and $r_y^{(l)}$, while code-ID nodes are initialized with the corresponding codebook vectors $q_{id}^{(l)}$, which act as prototypes of latent semantic clusters.

Residual Graph Aggregation. For each graph $\mathcal{G}^{(l)}$, we apply a K-layer Graph Attention Network (GAT) (Velickovic et al., 2018) to propagate information across case, charge, and code-ID nodes.

Let $h_v^{(l,\text{init})}$ denote the initial feature of node v in layer l, which is either the HSE vector $r_v^{(l)}$ for case and charge nodes or the codebook vector $\mathcal{E}^{(l)}[id_v^{(l)}]$ for code-ID nodes. The initial state of the first

[2]Detailed analysis is provided in Appendix D.2.

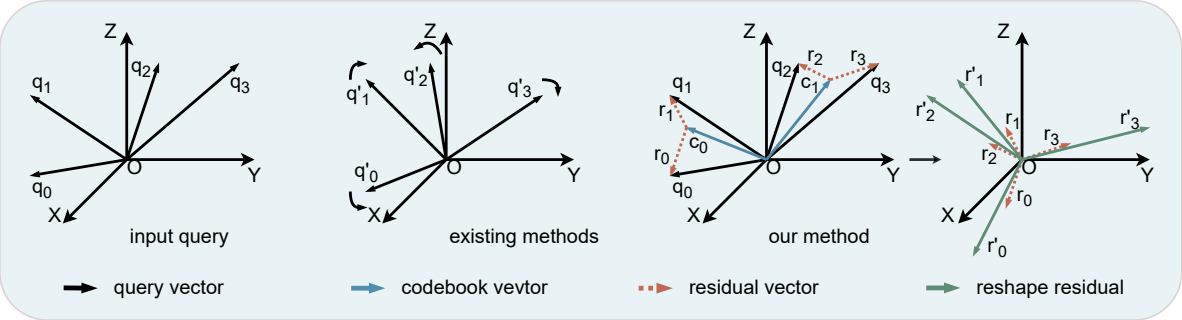

Figure 3: Illustration of query vector refinement. In this example, q_1 and q_2 are positive samples, while q_0 and q_3 are negatives. Compared with flat retrieval methods, our hierarchical design progressively separates negatives and groups positives while preserving multi-level semantic relations.

graph is

$$h_v^{(0,0)} = h_v^{(0,\text{init})}. \tag{7}$$

After graph aggregation on the l-th graph, the updated features of cases and charges are injected into the next graph through residual addition:

$$h_v^{(l+1,0)} = h_v^{(l+1,\text{init})} + h_v^{(l,K)}. \tag{8}$$

This preserves alignment of corresponding nodes across graphs while allowing deeper graphs to build on earlier semantic abstractions. Within each graph, for $k \in \{0, 1, \ldots, K-1\}$, node representations are updated as

$$h_v^{(l,k+1)} = \text{GAT}\left(h_v^{(l,k)}, \{h_u^{(l,k)} : u \in \mathcal{N}^{(l)}(v)\}\right), \tag{9}$$

where $\mathcal{N}^{(l)}(v)$ is the neighbor set of node v in graph $\mathcal{G}^{(l)}$. After K rounds of message passing, we obtain the case representation $g_i^{(l)} = h_i^{(l,K)}$ for the l-th graph.

Each graph aggregates information from semantically related cases linked by semantic IDs, yielding context-aware representations at the corresponding semantic level. The final representation of case c_i is formed by concatenating all layer-wise outputs with the original compact embedding e_i:

$$\hat{g}_i = [g_i^{(0)} \| g_i^{(1)} \| \ldots \| g_i^{(L-1)} \| e_i]. \tag{10}$$

This yields a rich case representation that preserves both coarse legal concepts and fine-grained factual distinctions.

4.3 Training and Inference

We adopt a two-stage training strategy. In the first stage, HSE learns a latent legal hierarchy and corresponding discrete semantic IDs. In the second stage, HSE is frozen, and CRL is trained to produce the final case representation $\hat{g}_i$.

For HSE, we aim to learn discriminative discrete representations rather than reconstructing inputs. At each quantization layer l, we use a contrastive objective:

$$\mathcal{L}_{\text{cl}}^{(l)} = \sum_{(a,p,n)} \text{ReLU}(\text{sim}(a,n) - \text{sim}(a,p) + m), \tag{11}$$

where (a, p, n) denotes an anchor, positive, and hard negative triplet, and hard negatives are selected from top-k neighbors in PCA-whitened space. To align quantized codes with residual inputs, we apply a codebook commitment loss:

$$\mathcal{L}_{\text{cb}}^{(l)} = \|\text{sg}[q_i^{(l)}] - r_i^{(l)}\|_2^2 + \beta \|q_i^{(l)} - \text{sg}[r_i^{(l)}]\|_2^2. \tag{12}$$

The overall HSE objective is defined as:

$$\mathcal{L}_{\text{HSE}} = \sum_{l=0}^{L-1} \left(\gamma^l \mathcal{L}_{\text{cl}}^{(l)} + \mathcal{L}_{\text{cb}}^{(l)}\right), \tag{13}$$

where $\gamma > 1$ is a hyperparameter.

For CRL, we apply an InfoNCE loss (van den Oord et al., 2018) on the final representation $\hat{g}_i$, using the same sampling strategy as HSE. GAT parameters are shared across all graphs $\mathcal{G}^{(l)}$, reducing model size and enforcing consistent message passing across semantic levels.

During inference, a query is encoded through the same pipeline, and cases are ranked by cosine similarity with $\hat{g}_i$. Since GAT aggregation solely depends on local K-hop neighborhoods, we construct only the corresponding subgraph for inference, enabling efficient retrieval while preserving multi-level semantics.

Theoretical Insight. As shown in Fig. 3, our hierarchical design progressively separates hard negatives while preserving semantic relationships

across layers. Additionally, this design encourages a coarse-to-fine semantic encoding, which is friendly to graph reasoning:

Proposition 1 *Under the loss-scaling scheme of DSLaw, the most distinguishing semantic information tends to concentrate in the final codebook.*

The proof is provided in Appendix B. The analyses in Sec. 5.4 also verify this property.

5 Experiments

5.1 Experimental Settings

Datasets. We evaluate DSLaw on two anonymized public legal datasets covering both common-law and civil-law systems:

- **COLIEE2025** (Goebel et al., 2024): COLIEE2025 is an English dataset of historical Canadian case law. Its copyright is not fully open (Goebel et al., 2024), and permission can be obtained from the official website (`https://coliee.org/`). The corpus contains approximately 7,350 labeled cases with 1,678 queries in the training set and 2,159 cases with 400 queries in the test set. Each query is associated with a set of relevant cases, where relevance is defined by explicit citation links between cases. We use the official splits provided by the dataset organizers.
- **ELAM** (Yu et al., 2022): ELAM is a publicly available Chinese corpus of historical criminal cases, originally developed for entity-centric legal argument mining. It contains 2,364 cases and over 5,000 manually annotated case pairs, each labeled to indicate whether two cases are semantically relevant. Since ELAM does not provide official splits, we construct the test set by repeatedly sampling a case from the corpus (random seed = 42), adding this case together with all its related cases to the test set, and continuing until the target test-set size is reached. During evaluation, each query is matched against the entire corpus without a predefined candidate pool, enabling open-domain retrieval.

On complex legal benchmarks like MUSER (Li et al., 2023) or LeCaRDv2 (Li et al., 2024), DSLaw can align its hierarchical semantics with specific dimensions to capture multi-faceted structures.

Data Anonymization. Both datasets used in this work are anonymized public legal datasets. Identifiable personal information has been removed or replaced during dataset construction and preprocessing. In addition, when generating summaries in LSE, we instruct the LLM to use role-based expressions such as "plaintiff", "defendant", and "judge" rather than specific names. These practices preserve the legal semantics of the cases while reducing privacy risks. We use the datasets solely for research purposes.

Baselines. We compare DSLaw with representative baselines from three categories. Traditional retrieval methods include TF-IDF (Aizawa, 2003) and BM25 (Robertson and Zaragoza, 2009). LLM-based methods include LegalBERT (Chalkidis et al., 2020), Lawformer (Xiao et al., 2021), RankGPT (Sun et al., 2023), BGE (Xiao et al., 2024), BGE-Reranker (Chen et al., 2024), DELTA (Li et al., 2025), recent LAV (Meng et al., 2025) and GLIER (Li et al., 2026). Structure-aware methods include PromptCase (Tang et al., 2023), CaseLink (Tang et al., 2024), KELLER (Deng et al., 2024), GEAR (Qin et al., 2024), and LEXA (Tang et al., 2026). All embedding-based methods are implemented using Qwen3-Embedding-8B (Zhang et al., 2025).

Implementation. We adopt either official implementations or widely used Python libraries (Lin et al., 2021) for traditional retrieval methods. For trainable baselines, we follow the settings used in GEAR (Qin et al., 2024), truncating longer inputs when necessary. For DSLaw, we use a 6-layer residual quantizer with embedding dimension 512 and 64 entries per codebook. Each graph employs a 2-layer GAT. The learning rate is set to 10^{-3} with Adam and a warm-up ratio of 0.1. Unless otherwise specified in the original papers, all trainable methods use a batch size of 256 and early stopping. All methods share the same pretrained embedding model for initial text representations. Appendix C provides more implementation details of DSLaw.

Environment. We use a machine with two Intel Xeon Silver 4314 CPUs, 128 GB main memory, and one NVIDIA A800 GPU.

5.2 Overall Performance[3]

Tab. 1 reports the overall performance:

- Both DSLaw and its joint-training variant achieve the best overall performance on COLIEE2025 and ELAM, demonstrating the effectiveness of DSLaw across different legal systems. Notably, joint training on the unified corpus fur-

[3]Complete results and broader ablation studies are provided in Appendix D.1.

	COLIEE2025					ELAM				
Model Name	R@5	N@5	P@5	MRR	MAP	R@5	N@5	P@5	MRR	MAP
Traditional Retrieval Methods										
TFIDF	0.2296	0.1793	0.1515	0.2317	0.1554	0.0242	0.0232	0.0260	0.0541	0.0178
BM25	0.2315	0.2492	0.1680	0.3978	0.2187	0.0672	0.0660	0.0507	0.1511	0.0769
LLM-based Methods										
LegalBERT	0.0510	0.0437	0.0415	0.0828	0.0343	0.0069	0.0052	0.0056	0.0119	0.0035
RankGPT	0.2089	0.2140	0.1535	0.3485	0.1911	0.0665	0.0689	0.0559	0.1667	0.0690
BGE-Reranker	0.2018	0.2046	0.1440	0.3220	0.1809	0.0711	0.0606	0.0520	0.1440	0.0676
DELTA	0.1636	0.1667	0.1215	0.2735	0.1432	0.0503	0.0518	0.0441	0.1351	0.0723
LVA	0.1932	0.1879	0.1315	0.3168	0.1792	0.0794	0.0833	0.0689	0.1643	0.0761
Structure-aware Methods										
PromptCase	0.2301	0.2430	0.1755	0.3842	0.2292	0.0582	0.0631	0.0475	0.1632	0.0811
CaseLink	0.2354	0.2402	0.1760	0.3706	0.2351	0.0261	0.0283	0.0282	0.0852	0.0505
LEXA	0.2503	0.2576	0.1812	0.4017	0.2416	0.0328	0.0544	0.0372	0.0972	0.0411
KELLER	0.0551	0.0591	0.0410	0.1091	0.0499	0.0669	0.0677	0.0520	0.1677	0.0681
GEAR	–	–	–	–	–	0.0988	0.0971	0.0807	0.2048	0.0863
GLIER	0.0931	0.0890	0.0684	0.1922	0.0816	0.1006	0.1023	0.0773	0.2102	0.0834
DSLaw	0.2874	0.2749	0.1880	0.4071	0.2446	0.1110	0.1085	0.0854	0.2201	0.1012
DSLaw (joint)	**0.2892**	**0.2778**	**0.1905**	**0.4163**	**0.2478**	**0.1124**	**0.1137**	**0.0887**	**0.2217**	**0.1063**

Table 1: Main results on COLIEE2025 and ELAM. "R", "N", and "P" denote Recall, NDCG, and Precision, respectively. GEAR is not evaluated on COLIEE2025 because it requires structural features unavailable in common-law systems. **Bold** values indicate the best results, and underlined values denote the strongest baseline results. Joint training combines the training sets of COLIEE2025 and ELAM into a unified training corpus. Both DSLaw and its joint-training variant significantly outperform the baselines under the same experimental settings (two-tailed paired t-test with Bonferroni correction, $p < 0.05$).

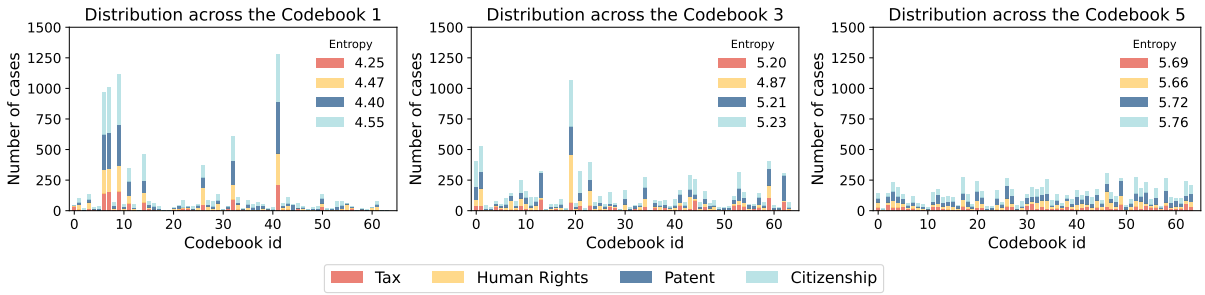

Figure 4: Distribution of legal cases across different codebooks (Codebook 1, 3, 5), showing the number of cases assigned to each codebook ID with different legal charges. The colors represent different legal charges, and the entropy values at the top indicate the level of dispersion across the codebook IDs.

ther improves performance, suggesting that the proposed framework can benefit from shared semantic structure across legal systems.

- Traditional retrieval methods such as BM25 remain highly competitive in LCR, especially on COLIEE2025, while general pretrained language models show only limited effectiveness. This suggests that lexical overlap still provides strong retrieval signals, whereas LLM-based dense retrieval methods are insufficient for capturing fine-grained legal distinctions.

- Structure-aware models perform well within specific legal domains but generalize poorly across legal systems. CaseLink is competitive in the common-law setting but drops substantially on civil-law data, whereas GEAR achieves strong results on ELAM but is not applicable to COLIEE2025. This highlights the limitation of methods tailored to system-specific legal structures and further demonstrates the advantage of DSLaw as a unified retrieval framework.

These results indicate that DSLaw provides a unified framework for LCR, in which both training and inference can be conducted consistently across legal systems, and cross-system corpora can be

	COLIEE2025		ELAM	
Setting	R@5	N@5	R@5	N@5
DSLaw (Qwen)	0.2874	0.2749	0.1110	0.1085
DSLaw (BGE)	0.2823	0.2703	0.1118	0.1092
w/o LSE	0.2603	0.2633	0.0752	0.0784
w/o HSE	0.2732	0.2676	0.0683	0.0788
w/o CRL	0.2686	0.2717	0.0680	0.0792
w/o ALL	0.1181	0.1108	0.0669	0.0691

Table 2: Ablation and backbone replacement results of DSLaw on COLIEE2025 and ELAM.

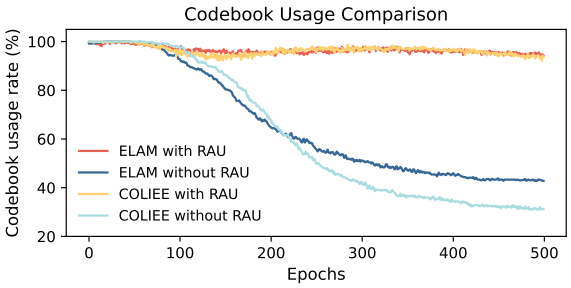

Figure 5: Codebook usage rate with and without RAU across all codebooks.

jointly exploited to learn a single retrieval model applicable to multiple legal traditions.

5.3 Ablation Studies[3]

To evaluate the contribution of each component in DSLaw, we remove LSE, HSE, and CRL individually, as well as removing all three components. The results are reported in Tab. 2.

Removing any component leads to consistent performance drops on both COLIEE2025 and ELAM, confirming that each module contributes to the final retrieval quality. The largest decline is observed in the "w/o ALL" setting, where DSLaw degenerates to direct retrieval over raw case embeddings. This result further verifies that the integration of preprocessing, hierarchical discrete semantics, and graph-based aggregation is essential to the overall effectiveness of DSLaw.

In the main experiments, DSLaw adopts Qwen3-Embedding-8B (Zhang et al., 2025) as the default embedding backbone. Replacing it with BGE-M3 (Xiao et al., 2024) yields highly consistent results with only minor variations, indicating that the proposed framework is robust to the choice of embedding model.

5.4 Analysis of Latent Legal Hierarchy

To examine whether HSE captures hierarchical legal semantics, we analyze the case distribution across codebooks in Fig. 4. Cases with similar legal labels are initially clustered closely in the representation space, indicating that the learned discrete IDs encode coarse semantic regularities. To capture finer-grained semantic distinctions, contrastive learning progressively separates hard negative samples that were originally close to each other. As the hierarchy deepens, these initially compact clusters spread across more codebook entries, leading to increasing entropy in deeper codebooks.

This trend shows that cases initially grouped under the same broad category become progressively more separable in later stages, demonstrating that the latent legal hierarchy captures semantic features at different levels of granularity, from coarse legal concepts to fine-grained factual distinctions. Additional visualizations are provided in Appendix D.2.

5.5 Analysis of the RAU Module

In conventional residual quantization, deeper residuals are obtained by direct subtraction from previous layers. As a result, downstream discrete IDs are largely constrained by the initial encoder output, and residual magnitudes progressively diminish, reducing discriminability in deeper codebooks. RAU addresses this issue by introducing a nonlinear transformation before residuals are propagated to the next layer.

To evaluate its effect, we compare the codebook usage rate with and without RAU in Fig. 5, where the usage rate is defined as the proportion of activated entries in each codebook. DSLaw with RAU maintains a consistently higher usage rate throughout training, whereas removing RAU leads to a rapid decline in activated entries and eventual codebook saturation. This indicates that RAU stabilizes hierarchical quantization, mitigates codebook collapse, and yields more diverse and expressive representations.

5.6 Case Study

To intuitively illustrate how DSLaw works, we present a representative example with three cases from the test set[4]. Case 1 is the query, Case 2 is the correct precedent, and Case 3 is a hard negative. Although all three cases arise from a similar labor-law context, their legal issues differ at a finer level: Cases 1 and 2 concern compensation calculations for work on scheduled rest days, whereas Case 3 concerns scheduling disputes under a collective

[4]Detailed case descriptions are provided in Appendix D.3.

agreement.

The keywords of the three cases are as follows:

- **Case 1:** [Labour, Transportation, Patent Unreasonable, Judicial Review, Costs]
- **Case 2:** [Labour, Human Rights, Patent Unreasonable, Judicial Review]
- **Case 3:** [Labour, Tax, Administrative Law, Judicial Review, Costs]

Standard BM25 incorrectly ranks Case 3 above Case 2 because Case 1 and Case 3 share stronger lexical overlap in procedural expressions such as "Judicial Review" and "Costs". Direct dense embeddings also fail to distinguish the subtle legal issue, assigning a higher similarity to the hard negative than to the true precedent (Case 2: 0.658 vs. Case 3: 0.718).

LSE reduces this mismatch by extracting more focused legal phrases, such as "Patent Unreasonable," which better reflect the underlying legal issue. After summarization, the similarity gap narrows (Case 2: 0.692 vs. Case 3: 0.725), but the ranking is still not corrected.

HSE then produces the following discrete ID sequences:

- **Case 1:** `[11, 48, 9, 37, 3, 6]`
- **Case 2:** `[11, 48, 9, 45, 14, 6]`
- **Case 3:** `[11, 48, 9, 37, 28, 11]`

The shared shallow IDs `[11,48,9]` place all three cases in the same broad legal domain, while the deeper layers expose the key factual distinction. In particular, Cases 1 and 2 align on the final ID `6`, whereas Case 3 diverges to `11`. This shows that HSE captures not only broad legal similarity, but also the fine-grained factual nuance relevant for precedent retrieval.

Finally, CRL amplifies this signal by connecting Cases 1 and 2 through the shared deep ID, which acts as a semantic bridge in the graph. The final scores reverse the incorrect ranking, with Case 2 correctly ranked above Case 3 (Case 2: 0.784 vs. Case 3: 0.736). This example shows how DSLaw progressively moves from lexical matching, to summary-enhanced representation, to hierarchical semantics, and finally to graph-based relational reasoning.

5.7 Additional Analyses

On the enlarged one-million-node graph, full-graph inference requires 1.5928 s per query and 77.84 GB of peak GPU memory, whereas query-centered 2-hop local inference reduces these costs to 0.1918 s per query and 9.29 GB, respectively. CPU memory consumption is also reduced from 3.00 GB to 1.24 GB. These results demonstrate that query-centered local inference substantially reduces the computational and memory overhead of graph reasoning at large scales. Additional analyses, including efficiency analysis, parameter sensitivity analysis, and stratified semantic analysis across charge types, are provided in Appendices D.4, D.5, and D.6, respectively.

6 Conclusion

DSLaw models legal cases through a latent legal hierarchy, where RQ generates layered discrete IDs that capture semantic distinctions ranging from coarse legal concepts to fine-grained factual details. These discrete IDs further enable hierarchical graph-based reasoning. By organizing legal cases through shared hierarchical semantics rather than system-specific structures, DSLaw provides a foundation for unified LCR across legal systems. Future work may involve aligning the implicit legal hierarchy constructed by DSLaw with explicitly defined legal meanings.

Limitations

Although DSLaw provides a unified framework for LCR across legal systems, its latent legal hierarchy is learned implicitly from case representations rather than aligned with explicit legal taxonomies or expert-defined doctrinal structures. As a result, the learned discrete IDs can already provide a certain degree of interpretability, but their legal meaning is still induced from data rather than explicitly defined. More comprehensive interpretability may therefore require further study of semantic IDs, especially their finer-grained legal meaning and possible alignment with explicit legal knowledge.

Acknowledgements

This work is supported by the New Generation Artificial Intelligence-National Science and Technology Major Project (No. 2025ZD0122702), National Natural Science Foundation of China (No. 62572410, No. 62272401, No. 62602536), Natural Science Foundation of Xiamen, China (No. 3502Z202471028), and the Fundamental Research Funds for the Central Universities, Xiamen University (No. 20720250171).

References

Akiko N. Aizawa. 2003. An information-theoretic perspective of tf-idf measures. *Inf. Process. Manag.*, 39(1):45–65.

Andres Buzo, Augustine H. Gray Jr., Robert M. Gray, and John D. Markel. 1980. Speech coding based upon vector quantization. In *ICASSP*, pages 15–18.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: the muppets straight out of law school. *arXiv Preprint*. https://arxiv.org/abs/2010.02559.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv Preprint*. https://doi.org/10.48550/arXiv.2402.03216.

Yongjian Chen, Tao Guan, and Cheng Wang. 2010. Approximate nearest neighbor search by residual vector quantization. *Sensors*, 10(12):11259–11273.

Chenlong Deng, Kelong Mao, and Zhicheng Dou. 2024. Learning interpretable legal case retrieval via knowledge-guided case reformulation. In *EMNLP*, pages 1253–1265.

Gregor Donabauer and Udo Kruschwitz. 2025. A reproducibility study of graph-based legal case retrieval. In *SIGIR*, pages 3135–3144.

Yi Feng, Chuanyi Li, and Vincent Ng. 2024. Legal case retrieval: A survey of the state of the art. In *ACL*, pages 6472–6485.

Randy Goebel, Yoshinobu Kano, Mi-Young Kim, Juliano Rabelo, Ken Satoh, and Masaharu Yoshioka. 2024. Overview and discussion of the competition on legal information, extraction/entailment (COLIEE) 2023. *Rev. Socionetwork Strateg.*, 18(1):27–47.

Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. 2023. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *ICML*, volume 202, pages 14096–14113.

Haitao Li, Qingyao Ai, Xinyan Han, Jia Chen, Qian Dong, and Yiqun Liu. 2025. DELTA: pre-train a discriminative encoder for legal case retrieval via structural word alignment. In *AAAI*, pages 27072–27080.

Haitao Li, Yunqiu Shao, Yueyue Wu, Qingyao Ai, Yixiao Ma, and Yiqun Liu. 2024. Lecardv2: A large-scale chinese legal case retrieval dataset. In *SIGIR*, pages 2251–2260.

Minghan Li, Tianrui Lv, Chao Zhang, and Guodong Zhou. 2026. GLIER: Generative legal inference and evidence ranking for legal case retrieval. In *ACL*, pages 29561–29572.

Qingquan Li, Yiran Hu, Feng Yao, Chaojun Xiao, Zhiyuan Liu, Maosong Sun, and Weixing Shen. 2023. MUSER: A multi-view similar case retrieval dataset. In *CIKM*, pages 5336–5340.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *SIGIR*, pages 2356–2362.

Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tie-Yan Liu, and Arnold Overwijk. 2021. Less is more: Pretrain a strong siamese encoder for dense text retrieval using a weak decoder. In *EMNLP*, pages 2780–2791.

Chunyun Meng, Cheng Tang, Yuki Todo, and Weiping Ding. 2025. Multi-granular legal information fusion with adversarial compensation: A hierarchical and logic-aware framework for robust case retrieval. *Knowledge-Based Systems*, 325:113964.

Weicong Qin, Zelin Cao, Weijie Yu, Zihua Si, Sirui Chen, and Jun Xu. 2024. Explicitly integrating judgment prediction with legal document retrieval: A law-guided generative approach. In *SIGIR*, pages 2210–2220.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Mahesh Sathiamoorthy. 2023. Recommender systems with generative retrieval. In *NeurIPS*.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Yunqiu Shao, Yueyue Wu, Yiqun Liu, Jiaxin Mao, and Shaoping Ma. 2023. Understanding relevance judgments in legal case retrieval. *ACM Trans. Inf. Syst.*, 41(3):76:1–76:32.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *EMNLP*, pages 14918–14937.

Yanran Tang, Ruihong Qiu, and Xue Li. 2023. Prompt-based effective input reformulation for legal case retrieval. In *ADC*, volume 14386, pages 87–100.

Yanran Tang, Ruihong Qiu, Yilun Liu, Xue Li, and Zi Huang. 2026. Lexa: Legal case retrieval via graph contrastive learning with contextualised llm embeddings. *World Wide Web*, 29:20.

Yanran Tang, Ruihong Qiu, Hongzhi Yin, Xue Li, and Zi Huang. 2024. Caselink: Inductive graph learning for legal case retrieval. In *SIGIR*, pages 2199–2209.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv Preprint*. https://arxiv.org/abs/1807.03748.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *NIPS*, pages 6306–6315.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*. https://openreview.net/forum?id=rJXMpikCZ.

Chaojun Xiao, Xueyu Hu, Zhiyuan Liu, Cunchao Tu, and Maosong Sun. 2021. Lawformer: A pre-trained language model for chinese legal long documents. *AI Open*, 2:79–84.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *SIGIR*, pages 641–649.

Weijie Yu, Zhongxiang Sun, Jun Xu, Zhenhua Dong, Xu Chen, Hongteng Xu, and Ji-Rong Wen. 2022. Explainable legal case matching via inverse optimal transport-based rationale extraction. In *SIGIR*, pages 657–668.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2022a. Soundstream: An end-to-end neural audio codec. *IEEE ACM Trans. Audio Speech Lang. Process.*, 30:495–507.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2022b. Soundstream: An end-to-end neural audio codec. *IEEE ACM Trans. Audio Speech Lang. Process.*, 30:495–507.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv Preprint*. https://arxiv.org/abs/2506.05176v2.

A Details of Legal Summary Extraction

Let $\mathcal{C} = \{c_0, c_1, \ldots, c_{N-1}\}$ denote a corpus of legal case documents, where each case c_i consists of raw text $\mathcal{T}_i$ and a set of charge labels $y_i \subseteq \mathcal{Y}$, with $\mathcal{Y}$ being the set of all charge categories. We first split each legal document $\mathcal{T}_i$ into sentences $\{s_0, s_1, \ldots, s_{n_i-1}\}$. For each sentence s_j, we compute a combined importance score:

$$\text{Score}(s_j) = \alpha \cdot T(s_j) + (1-\alpha) \cdot K(s_j), \quad (14)$$

where $T(s_j)$ denotes the average TF-IDF score of sentence s_j, and $K(s_j)$ denotes the number of matches with a predefined set of legal keywords. The hyperparameter $\alpha \in [0, 1]$ balances these two components. In our experiments, we set $\alpha = 0.5$.

We then concatenate the top-K ranked sentences together with all matched legal keywords of c_i to form the input to the LLM. We set $K = 10$ for COLIEE2025 and $K = 5$ for ELAM.

Prompt for summarization. Given the extracted keywords and text excerpts, we use the following prompt to generate a concise summary $\tilde{\mathcal{T}}_i$:

> You will be given a set of keywords and multiple text excerpts related to a legal case. Your task is to concisely summarize the legal proceedings, emphasizing the key legal concepts, relevant facts, judicial reasoning, decisions, and final judgment.
>
> **Keywords:** {<keywords>}
> **Text 1:** {<excerpt 1>}
> **Text 2:** {<excerpt 2>}
> **...**
> **Text k:** {<excerpt k>}
>
> Your summary should be logically structured, clear, and no more than 200 words. Avoid introductory phrases such as "Here is the summary." Use general legal terms such as "plaintiff," "defendant," "the company," and "the judge" instead of specific names. Focus on the core legal aspects and procedural developments without unnecessary details or commentary. Do not include any additional explanations or annotations.

We use Qwen3-8B to generate the summaries.

Embedding and whitening. The generated summary $\tilde{\mathcal{T}}_i$ is encoded into a dense representation $e'_i \in \mathbb{R}^{d'}$ using Qwen3-Embedding-8B, which is selected for its robust multilingual capabilities required to align cross-system legal documents into a unified latent space. We then apply PCA followed by whitening:

$$e_i = \text{Whiten}\big(\text{PCA}(e'_i)\big), \quad (15)$$

where the PCA transformation is estimated on the training set. We retain 95% of the variance in the

reduced space. Each charge label $y \in \mathcal{Y}$ is encoded using the same embedding model and PCA-whitening transformation, yielding a charge embedding $e_y \in \mathbb{R}^d$.

B Theoretical Analyses of Codebook Design

This appendix provides the theoretical analyses of Proposition 1 in Sec. 4.3, i.e., under our loss-scaling scheme, the most distinguishing semantic information tends to concentrate in the final codebook. For clarity, the discussion is divided into three parts.

- **B.1:** We analyze a simplified setting in which input features are assumed to be independent and exchangeable (no ordering).
- **B.2:** We then relax the independence assumption and consider statistical dependencies among input features.
- **B.3:** Finally, we discuss the theoretical bounds of the scaling factor γ.

B.1 Independent Features (No Information Bottleneck)

Setting. For each layer $l \in \{0, \ldots, L-1\}$ define the weighted contrastive loss

$$\tilde{L}_{\text{cl}}^{(l)} \;=\; \gamma^{\,l}\, L_{\text{cl}}^{(l)}, \qquad \gamma > 1, \tag{16}$$

and the overall objective

$$L_{\text{HSE}} \;=\; \sum_{l=0}^{L-1} \Big(\tilde{L}_{\text{cl}}^{(l)} + L_{\text{cb}}^{(l)} \Big), \qquad L_{\text{cb}}^{(l)} \geq 0. \tag{17}$$

Assumptions. The semantic components extracted by different layers are *mutually independent*: each layer operates in a sub-space whose discriminative information does not rely on any specific preprocessing performed by earlier layers. Hence reducing a shallower $L_{\text{cl}}^{(l)}$ can *never increase* any deeper $L_{\text{cl}}^{(k)}$ $(k > l)$.

Claim. Every (local) minimiser of (17) satisfies

$$L_{\text{cl}}^{(0)} \;\geq\; L_{\text{cl}}^{(1)} \;\geq\; \ldots \;\geq\; L_{\text{cl}}^{(L-1)}, \tag{18}$$

so the deepest codebook ($l = L-1$) attains the smallest contrastive loss, which means that the most distinguishing semantic information is captured in the final codebook.

Proof by contradiction. Assume that at a stationary point there exists an index $m \in \{0, \ldots, L-2\}$ such that

$$L_{\text{cl}}^{(m+1)} \;>\; L_{\text{cl}}^{(m)}. \tag{19}$$

Construct a swapped model. Because every layer shares the same architecture, dimensionality, code-vector cardinality, and—by the independence assumption—acts on an independent feature subspace, we can swap the entire parameter sets of layers m and $m+1$ without altering the validity of the network forward path. All other layers remain unchanged, and every code-commitment term $L_{\text{cb}}^{(l)}$ is therefore identical before and after the swap. The change in the objective (17) is solely due to the two weighted contrastive terms:

$$\begin{aligned} \Delta &= \big[\gamma^{\,m} L_{\text{cl}}^{(m+1)} + \gamma^{\,m+1} L_{\text{cl}}^{(m)}\big] - \big[\gamma^{\,m} L_{\text{cl}}^{(m)} + \\ &\quad \gamma^{\,m+1} L_{\text{cl}}^{(m+1)}\big] \end{aligned} \tag{20}$$

$$= \big(\gamma^{\,m+1} - \gamma^{\,m}\big)\big(L_{\text{cl}}^{(m)} - L_{\text{cl}}^{(m+1)}\big) < 0, \tag{21}$$

where the strict inequality follows from (19) and $\gamma^{\,m+1} > \gamma^{\,m}$.

Since $\Delta < 0$, the swapped model attains a strictly smaller value of L_{HSE}, contradicting the local optimality of the original point. Therefore the assumption (19) is impossible, and (18) must hold. □

B.2 Dependent Features (Information Bottleneck Allowed)

Setting. We reuse definitions (16)–(17).

Assumptions. The mutual-independence assumption is **dropped**: deeper layers may depend on earlier layers to perform a mandatory *information-bottleneck* transformation before new discriminative factors become separable. Consequently the raw losses $L_{\text{cl}}^{(l)}$ are allowed to rise and fall along the depth.

Claim. Every (local) minimiser of (17) satisfies

$$L_{\text{cl}}^{(L-1)} \;=\; \min_{0 \leq l \leq L-1} L_{\text{cl}}^{(l)}, \tag{22}$$

i.e. the *unweighted* contrastive loss attains its smallest value at the final layer, even in the presence of an information bottleneck. It means that the most distinguishing semantic information is captured in the final codebook.

Proof by contradiction. Assume there exists a local minimiser for which

$$L_{\text{cl}}^{(m)} \;<\; L_{\text{cl}}^{(L-1)} \qquad \text{for some } m < L-1. \tag{23}$$

Degenerating the deeper layers. Because every layer shares the same architecture and code-book size, the parameter sets of layers $m+1, \dots, L-1$ can be replaced by *identity mappings*: each such layer simply copies the discrete index produced by layer m. This modification (i) leaves all code-commitment penalties unchanged ($L_{\text{cb}}^{(l)} \geq 0$), and (ii) guarantees

$$L_{\text{cl,new}}^{(l)} \;\leq\; L_{\text{cl}}^{(m)}, \qquad m < l \leq L-1, \tag{24}$$

because an identity mapping can replicate, and often improve upon, the separability already achieved by layer m.

Change in the objective. With (24) the objective variation is bounded by

$$\begin{aligned}\Delta &= \sum_{l=m+1}^{L-1} \gamma^{\,l}\big(L_{\text{cl,new}}^{(l)} - L_{\text{cl}}^{(l)}\big) \\ &\leq \sum_{l=m+1}^{L-1} \gamma^{\,l}\big(L_{\text{cl}}^{(m)} - L_{\text{cl}}^{(l)}\big) \\ &< 0,\end{aligned} \tag{25}$$

where the strict inequality uses $\gamma^{\,l} > \gamma^{\,m}$ for $l > m$ together with the assumption (23) that at least layer $L-1$ has a larger loss than layer m.

Contradiction. Since $\Delta < 0$, the degenerate model achieves a strictly smaller value of L_{HSE}, contradicting the local minimality of the original point. Therefore (23) is impossible, and the proposition (22) holds.

□

B.3 The Impact of γ on Optimization

Setting. We reuse the definitions from (16)–(17).

Assumptions. We follow the assumptions in Proof 2. In addition, we assume that the model is not currently at a local minimum but at point S, and there exist two local minima, denoted as A and B, whose existence is derived from Proof B.2. Specifically:

- **At point A:** After layer m, all subsequent layers degrade to identity mappings (i.e., the output of layer $m+1$ through layer $L-1$ is a copy of the output from layer m).
- **At point B:** After layer n, all subsequent layers degrade to identity mappings (i.e., the output of layer $n+1$ through layer $L-1$ is a copy of the output from layer n).

We further assume that $m < n$, $L_{\text{cl}}^{(m)} > L_{\text{cl}}^{(n)}$, and that the loss at points A and B satisfy:

$$L_{\text{HSE}}^{A} < L_{\text{HSE}}^{B} \quad \text{when} \quad \gamma = 1. \tag{26}$$

Claim. As γ increases, the model is more likely to converge to the deeper local minimum at B, thus avoiding the shallower local minimum at A.

1. With $\gamma = 1$ (no scaling): Define the loss between layers $m+1$ and $n-1$ as:

$$L_{m+1}^{n-1} = \sum_{l=m+1}^{n-1} L_{\text{cl}}^{(l)}. \tag{27}$$

From the equation (26) and the assumption $L_{\text{cl}}^{(m)} > L_{\text{cl}}^{(n)}$, we have:

$$L_{\text{cl}}^{(m)} < \frac{1}{n-m-1} L_{m+1}^{n-1}. \tag{28}$$

The gradient with respect to the loss for each layer l in this range is given by $\nabla L_{\text{cl}}^{(l)}$, and the total gradient for the sum of these losses is:

$$\nabla\left(\sum_{l=m+1}^{n-1} L_{\text{cl}}^{(l)}\right) = \sum_{l=m+1}^{n-1} \nabla L_{\text{cl}}^{(l)}. \tag{29}$$

The direction of this gradient is pointing towards the direction of $S \to A$, meaning that the gradients from layers $m+1$ to $n-1$ contribute to minimizing the loss in a way that encourages convergence to the local minimum at A (the angle between the gradient direction and the path $S \to A$ is less than 90 degrees).

On the other hand, the direction from $S \to B$ here is random. When the direction from $S \to A$ is opposite to that of $S \to B$, the model has a higher probability of converging to the shallower minimum at A, as the gradient strongly favors convergence towards A rather than B.

Thus, the gradient direction points more strongly towards the shallow local minimum at A, making it more likely for the model to converge to A rather than the deeper minimum at B.

2. With $\gamma > 1$: When $\gamma > 1$, consider the gradient for layers from 1 to $n-1$. Suppose the maximum gradient occurs at layer p, i.e.,

$$\left|\nabla L_{\text{cl}}^{(p)}\right| = \max_{1 \leq l \leq n-1} \left|\nabla L_{\text{cl}}^{(l)}\right|. \tag{30}$$

Based on the chain rule for gradients, we have:

$$\nabla \left(\gamma^l L_{\text{cl}}^{(l)} \right) = \gamma^l \nabla L_{\text{cl}}^{(l)}. \tag{31}$$

Consider the total gradient for the layers from 1 to $n-1$:

$$\begin{aligned} \left| \sum_{l=1}^{n-1} \nabla \gamma^l L_{\text{cl}}^{(l)} \right| &\leq \left| \sum_{l=1}^{n-1} \gamma^l \nabla L_{\text{cl}}^{(p)} \right| \\ &= \mathcal{O}(r^{n-1}) \left| \nabla L_{\text{cl}}^{(p)} \right|. \end{aligned} \tag{32}$$

For the gradient at layer n, it can be written as:

$$\left| \nabla L_{\text{cl}}^{(n)} \right| = \mathcal{O}(\gamma^n) \left| \nabla L_{\text{cl}}^{(n)} \right|. \tag{33}$$

As γ increases, the gradient at the final layer n becomes the dominant term, and for sufficiently large γ, the gradient from the n-th layer will dominate the overall gradient. Specifically, when γ is large enough, the gradient at layer n increasingly determines the direction of the optimization process, causing the gradient direction to align more with $S \rightarrow B$ (the angle between two vectors is less than 90 degrees), and making it easier for the model to converge to the deeper minimum at B. This helps to better separate similar cases in subsequent layers.

At the same time, this ensures the overall stability of codebook's gradient, reducing the likelihood of catastrophic collapse in later codebook layers caused by drastic updates in the earlier layers.

C Implementation Details

The detailed implementations of HSE and CRL in DSLaw are described in the following:

C.1 Hierarchical Semantics Encoding

The HSE module first maps the input case embedding to a 512-dimensional space using a 2-layer MLP with a 768-unit hidden layer, ReLU activation, and dropout rate 0.3. We then apply a 6-layer residual quantization structure, where each codebook contains 64 entries and each layer has feature dimension 512.

Residual quantization configuration. At each layer, HSE selects the nearest codebook entry and progressively quantizes the residual representation. In implementation, we follow standard residual quantization practice and further adopt a lightweight codebook adaptation strategy following (Huh et al., 2023) to better align the quantized space with the layer-wise residual distribution. The detailed quantization equations are given in Sec. 3.

Training configuration. For training, the layer-wise contrastive loss is exponentially weighted with $\gamma = 2$. For each anchor, we select 30 hard negatives and set the contrastive margin to $m = 0.3$. The HSE module is optimized using Adam with learning rate 10^{-3} and weight decay 5×10^{-3}. Early stopping is applied when the validation loss does not improve for 50 consecutive epochs.

Residual Amplification Unit. For the Residual Amplification Unit (RAU), the hidden dimension is set to 128, the activation function is Swish, and LayerNorm is applied after the residual connection.

C.2 Case Representation Learning

In CRL, we construct a 6-layer graph hierarchy, where each layer uses a 2-layer Graph Attention Network (GAT). The first GAT layer uses 4 attention heads with hidden dimension 16, dropout rate 0.2, and ELU activation. The second GAT layer uses a single attention head with output dimension 512.

Graph propagation. Following recent findings that homogeneous message passing can be effective for LCR (Donabauer and Kruschwitz, 2025), we apply a homogeneous GAT over each graph $\mathcal{G}^{(l)}$, while preserving node-type information through type-specific linear projections on the initial node features.

Training and graph construction. For each anchor, we select 50 hard negatives and set the InfoNCE temperature to 0.1. The CRL module is trained with learning rate 10^{-3} for up to 1000 epochs.

To construct edges among charge nodes and among code-ID nodes, we adopt a dynamic thresholding strategy. At each layer, we select the top $M_{\text{charge}} = 10 \cdot |\mathcal{N}_{\text{charge}}|$ entries from the charge–charge similarity matrix and the top $M_{\text{id}} = 10 \cdot |\mathcal{N}_{\text{id}}|$ entries from the ID–ID similarity matrix. After removing duplicated symmetric pairs, the remaining pairs are treated as undirected edges. This keeps the graphs sufficiently sparse for efficient GAT computation while preserving useful semantic neighborhood structure.

For retrieval, we use cosine similarity. Therefore, all final case representations are L2-normalized before indexing.

D Supplementary Experiments

D.1 Complete Experimental Results

In this section, we provide the complete experimental results with all reported metrics. Tab. 3 and Tab. 4 present the detailed results on COLIEE2025, while Tab. 5 and Tab. 6 correspond to ELAM. In addition to the main model and its joint-training variant, we also report a broader set of ablation settings for completeness.

To evaluate the contribution of each component in DSLaw, we consider the following settings:

1. "w/ BGE-M3", where Qwen3-Embedding-8B is replaced by BGE-M3 as the embedding backbone in LSE;
2. "w/o LSE", which directly encodes the raw legal text using a pre-trained encoder;
3. "w/o HSE", which removes hierarchical discrete semantic modeling and replaces the layered discrete representations with repeated copies of the original embedding when constructing the multi-layer graphs;
4. "w/o CRL", which retains the 6-layer quantized representation but removes CRL, concatenating all quantized vectors with the original embedding for retrieval;
5. "w/o All", which removes all three modules and performs retrieval using Qwen3-Embedding-8B (Zhang et al., 2025) on the raw case texts;
6. "w/o RAU", which removes the Residual Amplification Unit between quantization layers in HSE;
7. "w/o CL", which replaces the contrastive objective in HSE with a reconstruction objective.

The complete results further confirm the trends reported in the main text. Across both datasets, DSLaw and its joint-training variant achieve the strongest overall performance, while the ablation settings show consistent degradations after removing individual components. In particular, the gains of joint training again suggest that the proposed framework can exploit shared semantic regularities across legal systems.

Results on COLIEE2025. As shown in Tab. 3 and Tab. 4, DSLaw achieves the strongest overall performance across nearly all evaluation metrics on COLIEE2025. Its joint-training variant further improves several metrics, which is consistent with the main-text observation that cross-system joint training can improve retrieval performance even on the common-law benchmark.

Structure-aware methods such as PromptCase and CaseLink remain relatively competitive on a subset of ranking metrics, but still fall short of DSLaw overall. In contrast, general dense retrieval models perform substantially worse, suggesting that semantic similarity alone is insufficient for citation-based legal case retrieval.

Results on ELAM. Tab. 5 and Tab. 6 report the complete results on ELAM. DSLaw again achieves the strongest overall performance across the major retrieval metrics, and its joint-training variant further improves the results on this civil-law dataset. This is consistent with the main finding that the proposed framework can benefit from joint training across legal systems.

Among the baselines, structure-aware models such as GEAR and PromptCase are more competitive than general dense retrieval models. However, these methods remain less stable across legal systems, whereas DSLaw maintains strong performance under a unified training and inference framework.

Additionally, we observe that the performance gains of DSLaw over baselines are generally more pronounced at larger k than at smaller k. This pattern is consistent with the nature of the task: at small k, retrieval is dominated by the highest-ranked candidates, where strong lexical overlap (as captured by methods like BM25) already provides reliable coarse-grained signals. As k increases, the model must distinguish a broader range of candidates with subtler legal differences, where the hierarchical discrete representations and graph-based reasoning of DSLaw provide a more significant advantage. In other words, the improvement brought by fine-grained semantic modeling becomes more apparent when the retrieval task requires covering a wider set of relevant cases rather than identifying only the single most obvious match.

D.2 Visualizations of Latent Legal Hierarchy

In DSLaw, shallow layers of the latent legal hierarchy tend to capture broad legal semantics, while deeper layers capture finer-grained signals that help distinguish hard negatives from anchors. Because COLIEE2025 associates multiple charges with each case whereas ELAM assigns a single charge per case, we use ELAM for visualization.

Fig. 6 shows representative clustering patterns

Model	R@1	R@5	R@10	R@20	R@30	N@1	N@5	N@10	N@20	N@30
TFIDF	*	0.2296	0.3347	0.4226	0.4959	*	0.1793	0.2228	0.2558	0.2798
BM25	0.0894	0.2315	0.3462	0.4398	0.4915	0.2800	0.2492	0.2871	0.3225	0.3397
LegalBERT	*	0.0510	0.0824	0.1290	0.1709	*	0.0437	0.0559	0.0738	0.0864
Lawformer	*	0.0199	0.0334	0.0671	0.0906	*	0.0190	0.0241	0.0370	0.0443
RankGPT	0.0702	0.2089	0.3195	0.4334	0.4990	0.2075	0.2140	0.2515	0.2942	0.3145
BGE	0.0266	0.0786	0.1244	0.1731	0.2022	0.1025	0.0876	0.1006	0.1203	0.1295
BGE-Reranker	0.0648	0.2018	0.3053	0.4033	0.4530	0.2050	0.2046	0.2390	0.2758	0.2915
DELTA	0.0480	0.1636	0.2456	0.3228	0.3819	0.1600	0.1667	0.1952	0.2246	0.2444
PromptCase	0.0772	0.2301	0.3304	0.4581	0.5489	0.2525	0.2430	0.2753	0.3225	0.3518
CaseLink	0.0694	0.2354	0.3528	0.4924	0.5566	0.2400	0.2402	0.2812	0.3319	0.3537
KELLER	0.0220	0.0551	0.0841	0.1186	0.1732	0.0600	0.0591	0.0692	0.0823	0.0975
DSLaw	0.1045	0.2874	0.3848	0.5083	0.5724	0.2925	0.2749	0.3013	**0.3420**	**0.3638**
DSLaw (joint)	**0.1045**	**0.2892**	**0.3976**	**0.5207**	**0.5892**	**0.2925**	**0.2778**	**0.3086**	0.3397	0.3624
DSLaw-w/ BGE-M3	0.1045	0.2823	0.3852	0.5067	0.5736	0.2925	0.2703	0.3072	0.3431	0.3641
DSLaw-w/o LSE	0.0928	0.2603	0.3621	0.4763	0.5348	0.2750	0.2633	0.2897	0.3276	0.3480
DSLaw-w/o HSE	0.0946	0.2732	0.3491	0.4554	0.5133	0.2725	0.2676	0.2967	0.3254	0.3417
DSLaw-w/o CRL	0.0856	0.2686	0.3579	0.4683	0.5208	0.2550	0.2717	0.2925	0.3403	0.3598
DSLaw-w/o ALL	0.0268	0.1181	0.1780	0.2721	0.3452	0.0780	0.1108	0.1336	0.1681	0.1906
DSLaw-w/o RAU	0.0906	0.2663	0.3680	0.4593	0.5191	0.2650	0.2631	0.2892	0.3367	0.3552
DSLaw-w/o CL	0.0741	0.1917	0.2463	0.3157	0.3582	0.1775	0.1459	0.1528	0.1685	0.1776

Table 3: Results on COLIEE2025. GEAR cannot be applied on COLIEE2025 because it leverages structural features that are absent in common-law systems. In the table, **bolded values** indicate the best performance for each metric, while underlined values denote the second-best results among the main methods. * indicates a negligible value (less than 0.0001). "R" and "N" stand for Recall and NDCG, respectively. DSLaw significantly outperforms baselines (two-tailed paired t-test with Bonferroni correction, $p < 0.05$) under the same experimental settings.

Model	P@1	P@5	P@10	P@20	P@30	ACC	MRR	MAP	mi–F1	ma–F1
TFIDF	*	0.1515	0.1223	0.0799	0.0641	*	0.2317	0.1554	0.1118	0.1422
BM25	0.2800	0.1680	0.1268	0.0841	0.0638	0.1100	0.3978	0.2187	0.1112	0.1387
LegalBERT	*	0.0415	0.0345	0.0279	0.0235	*	0.0828	0.0343	0.0410	0.0961
Lawformer	*	0.0205	0.0160	0.0151	0.0132	*	0.0462	0.0144	0.0230	0.0803
RankGPT	0.2075	0.1535	0.1162	0.0824	0.0633	0.0805	0.3485	0.1911	0.1103	0.1462
BGE	0.1025	0.0620	0.0445	0.0344	0.0269	0.0225	0.1722	0.0676	0.0470	0.0977
BGE-Reranker	0.2050	0.1440	0.1095	0.0759	0.0571	0.05	0.3220	0.1809	0.0996	0.1308
DELTA	0.1600	0.1215	0.0953	0.0646	0.0522	0.0525	0.2735	0.1432	0.0910	0.1346
PromptCase	0.2525	0.1755	0.1298	0.0916	0.0743	0.0875	0.3842	0.2292	0.1297	0.1489
CaseLink	0.2400	0.1760	0.1350	0.0967	0.0753	0.0825	0.3706	0.2351	0.1313	0.1559
KELLER	0.0600	0.0410	0.0327	0.0241	0.0212	0.0225	0.1091	0.0499	0.0369	0.0975
DSLaw	0.2925	0.1880	0.1453	**0.1058**	**0.0813**	0.1165	0.4071	0.2446	0.1425	0.1664
DSLaw (joint)	**0.2925**	**0.1905**	**0.1562**	0.1043	0.0801	**0.1192**	**0.4163**	**0.2478**	**0.1481**	**0.1692**
DSLaw-w/ BGE-M3	0.2925	0.1860	0.1472	0.1043	0.0796	0.1150	0.4067	0.2432	0.1435	0.1657
DSLaw-w/o LSE	0.2750	0.1815	0.1346	0.0872	0.0703	0.0915	0.4003	0.2276	0.2273	0.1472
DSLaw-w/o HSE	0.2725	0.1820	0.1253	0.0806	0.0615	0.0950	0.4032	0.2249	0.1073	0.1317
DSLaw-w/o CRL	0.2550	0.1840	0.1313	0.0906	0.0692	0.0850	0.4028	0.2343	0.1206	0.1439
DSLaw-w/o ALL	0.0780	0.0828	0.0669	0.0518	0.0435	0.0270	0.1923	0.0958	0.0759	0.1360
DSLaw-w/o RAU	0.2650	0.1820	0.1226	0.0801	0.0636	0.0938	0.3960	0.2283	0.1109	0.1496
DSLaw-w/o CL	0.1775	0.1250	0.0813	0.0531	0.0406	0.0625	0.3331	0.1285	0.0709	0.1252

Table 4: Results on COLIEE2025. GEAR cannot be applied on COLIEE2025 because it leverages structural features that are absent in common-law systems. In the table, **bolded values** indicate the best performance for each metric, while underlined values denote the second-best results among the main methods. * indicates a negligible value (less than 0.0001). DSLaw significantly outperforms baselines (two-tailed paired t-test with Bonferroni correction, $p < 0.05$) under the same experimental settings.

Model	R@1	R@5	R@10	R@20	R@30	N@1	N@5	N@10	N@20	N@30
TFIDF	*	0.0242	0.0467	0.0797	0.1059	*	0.0232	0.0329	0.0436	0.0518
BM25	0.0163	0.0672	0.1183	0.2221	0.3081	0.0601	0.0660	0.0862	0.1233	0.1506
LegalBERT	*	0.0069	0.0103	0.0152	0.0173	*	0.0052	0.0068	0.0089	0.0099
Lawformer	*	0.0270	0.0489	0.1062	0.1563	*	0.0220	0.0311	0.0510	0.0651
RankGPT	0.020	0.0665	0.1340	0.2459	0.3369	0.0650	0.0689	0.0953	0.1364	0.1653
BGE	0.0137	0.0573	0.1014	0.1742	0.2412	0.0454	0.0540	0.0722	0.0986	0.1193
BGE-Reranker	0.0133	0.0711	0.1267	0.2400	0.3089	0.0367	0.0606	0.0852	0.1250	0.1484
DELTA	0.0083	0.0503	0.1199	0.2229	0.2813	0.0424	0.0518	0.0825	0.1187	0.1378
PromptCase	0.0194	0.0582	0.1100	0.2053	0.2969	0.0763	0.0631	0.0828	0.1141	0.1423
CaseLink	0.0032	0.0261	0.0630	0.1360	0.2026	0.0169	0.0283	0.0433	0.0697	0.0904
KELLER	0.0240	0.0669	0.1346	0.2645	0.3501	0.0819	0.0677	0.0916	0.1353	0.1616
GEAR	0.0216	0.0988	0.1595	0.2762	0.3863	0.0682	0.0971	0.1207	0.1611	0.1939
DSLaw	0.0302	0.1110	0.1730	0.3318	0.4706	0.1017	0.1085	0.1299	0.1821	0.2215
DSLaw (joint)	**0.0296**	**0.1124**	**0.1812**	**0.3472**	**0.4923**	**0.1046**	**0.1137**	**0.1335**	**0.1933**	**0.2367**
DSLaw-w/ BGE-M3	0.0291	0.1118	0.1735	0.3278	0.4682	0.1013	0.1092	0.1307	0.1788	0.2173
DSLaw-w/o LSE	0.0173	0.0752	0.1296	0.2832	0.4436	0.0721	0.0784	0.0976	0.1458	0.1863
DSLaw-w/o HSE	0.0082	0.0683	0.1327	0.2981	0.4232	0.0626	0.0788	0.0912	0.1389	0.1776
DSLaw-w/o CRL	0.0111	0.0680	0.1249	0.2747	0.4304	0.0680	0.0792	0.0940	0.1346	0.1792
DSLaw-w/o All	0.0168	0.0669	0.1130	0.1980	0.2927	0.0593	0.0691	0.0849	0.1136	0.1409
DSLaw-w/o RAU	0.0266	0.0821	0.1486	0.3118	0.4436	0.0536	0.0732	0.1013	0.1479	0.1870
DSLaw-w/o CL	0.0155	0.0861	0.1401	0.3145	0.4279	0.0536	0.0779	0.1001	0.1548	0.1910

Table 5: Results on ELAM. In the table, **bolded values** indicate the best performance for each metric, while underlined values denote the second-best results among the main methods. * indicates a negligible value (less than 0.0001). "R" and "N" stand for Recall and NDCG, respectively. DSLaw significantly outperforms baselines (two-tailed paired t-test with Bonferroni correction, $p < 0.05$) under the same experimental settings.

Model	P@1	P@5	P@10	P@20	P@30	ACC	MRR	MAP	mi–F1	ma–F1
TFIDF	*	0.0260	0.0243	0.0198	0.0166	*	0.0541	0.0178	0.0294	0.0894
BM25	0.0601	0.0507	0.0462	0.0427	0.0398	0.0170	0.1511	0.0769	0.0705	0.1074
LegalBERT	*	0.0056	0.0045	0.0037	0.0030	*	0.0119	0.0035	0.0053	0.0667
Lawformer	*	0.0249	0.0226	0.0220	0.0200	*	0.0516	0.0212	0.0354	0.0898
RankGPT	0.0845	0.0559	0.0520	0.0481	0.0444	0.0065	0.1667	0.0690	0.0786	0.1252
BGE	0.0454	0.0415	0.0386	0.0338	0.0308	0.0153	0.1245	0.0590	0.0546	0.0964
BGE-Reranker	0.0508	0.0520	0.0497	0.0458	0.0412	0.0113	0.1440	0.0676	0.0730	0.1117
DELTA	0.0452	0.0441	0.0469	0.0441	0.0380	0.0113	0.1351	0.0723	0.0674	0.1031
PromptCase	0.0904	0.0475	0.0424	0.0376	0.0367	0.0282	0.1632	0.0811	0.0650	0.1014
CaseLink	0.0169	0.0282	0.0294	0.0302	0.0296	0.0056	0.0852	0.0505	0.0524	0.1032
KELLER	0.0904	0.0520	0.0486	0.0466	0.0433	0.0226	0.1677	0.0681	0.0767	0.1094
GEAR	0.0852	0.0807	0.0648	0.0551	0.0519	0.0114	0.2048	0.0863	0.0919	0.1246
DSLaw	0.1073	0.0854	0.0710	0.0602	0.0572	0.0326	0.2201	0.1012	0.1061	0.1358
DSLaw (joint)	**0.1086**	**0.0887**	**0.0742**	**0.0631**	**0.0583**	**0.0332**	**0.2217**	**0.1063**	**0.1108**	**0.1416**
DSLaw-w/ BGE-M3	0.1066	0.0866	0.0713	0.0587	0.0569	0.0318	0.2183	0.1014	0.1072	0.1337
DSLaw-w/o LSE	0.0733	0.0627	0.0599	0.0562	0.0551	0.0247	0.1478	0.0748	0.0972	0.1238
DSLaw-w/o HSE	0.0282	0.0554	0.0520	0.0559	0.0544	0.0056	0.1322	0.0673	0.0964	0.1222
DSLaw-w/o CRL	0.0621	0.0531	0.0514	0.0542	0.0535	0.0113	0.1508	0.0689	0.0947	0.1180
DSLaw-w/o All	0.0621	0.0565	0.0480	0.0398	0.0380	0.0169	0.1424	0.0568	0.0674	0.1025
DSLaw-w/o RAU	0.0767	0.0664	0.0599	0.0550	0.0531	0.0225	0.1621	0.0823	0.0974	0.1203
DSLaw-w/o CL	0.0621	0.0632	0.0571	0.0562	0.0537	0.0169	0.1598	0.0813	0.0951	0.1256

Table 6: Results on ELAM. In the table, **bolded values** indicate the best performance for each metric, while underlined values denote the second-best results among the main methods. * indicates a negligible value (less than 0.0001). DSLaw significantly outperforms baselines (two-tailed paired t-test with Bonferroni correction, $p < 0.05$) under the same experimental settings.

Case id	Discrete id	Charge	Case id	Discrete id	Charge
52	[26 17 58 27 8 26]	Providing Hacking Tools Crime	6	[62 1 53 4 59 46]	Using Fake or Stolen IDs
131	[26 17 58 41 17 3]	Providing Hacking Tools Crime	380	[62 1 53 4 59 46]	Using Fake or Stolen IDs
550	[26 17 58 5 17 3]	Providing Hacking Tools Crime	654	[62 1 53 38 59 55]	Using Fake or Stolen IDs
779	[26 17 58 38 17 23]	Providing Hacking Tools Crime	765	[62 1 53 25 53 7]	Forging or Trading IDs
986	[26 17 58 41 6 3]	Computer System Sabotage	767	[62 1 53 4 59 7]	Using Fake or Stolen IDs
1061	[26 17 58 1 22 3]	Providing Hacking Tools Crime	962	[62 1 53 4 59 36]	Using Fake or Stolen IDs
1321	[26 17 58 50 43 3]	Providing Hacking Tools Crime	1114	[62 1 53 4 59 55]	Using Fake or Stolen IDs
1567	[26 17 58 5 59 36]	Providing Hacking Tools Crime	1165	[62 1 53 4 59 55]	Using Fake or Stolen IDs
1653	[26 17 58 13 12 3]	Providing Hacking Tools Crime	1625	[62 1 53 40 59 46]	Using Fake or Stolen IDs
1883	[26 17 58 14 39 50]	Providing Hacking Tools Crime	1876	[62 1 53 4 59 60]	Using Fake or Stolen IDs
1940	[26 17 58 21 22 53]	Computer System Sabotage	2145	[62 1 53 19 37 7]	Using Fake or Stolen IDs
1946	[26 17 58 37 21 23]	Providing Hacking Tools Crime	2261	[62 1 53 4 59 50]	Forging or Trading IDs
943	[4 46 10 63 63 60]	Concealing or Transferring Drug	219	[14 23 38 24 29 7]	Illegal Assembly Organization
1052	[55 27 26 63 63 60]	Drug Smuggling and Trafficking	763	[60 58 39 24 29 7]	Assisting in Cybercrime
1575	[4 27 51 63 63 60]	Drug Smuggling and Trafficking	1682	[43 9 4 24 29 7]	Disrupting Social Order
2116	[12 8 4 63 63 60]	Illegal Possession of Drug			

Figure 6: Charge distribution for cases sharing identical first three or last three discrete IDs on ELAM.

after encoding:

- Cases that share the same first three discrete IDs are relatively numerous and mostly correspond to the same charge; even when the formal charges are not identical, they are often closely related to the dominant charge in the cluster. For instance, cases beginning with $\{26, 17, 58\}$ tend to cluster around cyber-related crimes, while those starting with $\{62, 1, 53\}$ relate to identity forgery.
- By contrast, cases that share the same last three discrete IDs are fewer, which is consistent with the entropy increase observed in Sec. 5.4, and their charge-level correlation becomes weaker. For example, cases with IDs 219, 763, and 1682 have different formal charges, but their fact patterns all involve providing financial support or assistance to others. This suggests that the deeper layers of latent legal hierarchy are not merely separating cases by charge labels, but are also capturing finer-grained factual regularities across related cases.

Overall, these visualizations provide further evidence that latent legal hierarchy organizes legal cases progressively from coarse semantic similarity to finer factual distinctions.

D.3 Case Study and Qualitative Analysis

To intuitively understand how DSLaw works, we present concrete examples with three cases from the test set. Case 1 (No.8510) is regarded as a query case, and its correct precedent needs to be found among cases 2 (No.864) and 3 (No.5296).

- **Case 1:** ... The applicants received summonses to attend a Public Service Labour Relations Board (PSLRB) adjudication hearing in relation to a co-worker. Each of the applicants was scheduled for a day of rest on October 18th. When they claimed overtime pay and expenses, the employer denied their claims... Based on article 23.01 of the Collective Agreement, the applicants claimed overtime pay and expenses with respect to their attendance at the hearing. The applicants submit that the adjudicator's decision was patently unreasonable because it is at odds with the plain and literal meaning of the words used in the article... According to the applicants, the word 'any' in article 23.01 refers to 'any' of the cases set out in article 30.17(c). In plain English, it cannot mean anything else... **Key words:** [Labour, Transportation, Patent Unreasonable, Judicial Review, Costs].
- **Case 2:** ... Under the lay-day operational manning system, the applicant Banton was normally scheduled to work a cycle of 28 consecutive days on board ship and 28 consecutive lay-days off, ashore. during his 28 scheduled lay-days off from December 25, 1991 to January 22, 1992, he was called in to work the last 14 days... He was compensated with his regular bi-weekly pay for the period from December 25, 1991 to January 22, 1992; and in addition, he received the cash equivalent of three lay-days for each of the 14 days worked... The applicant disputed the amount that he was paid for working on his scheduled lay-days. he took the position that the words 'earned lay-day pay'... meant that he should be compensated... at the rate of one lay-day plus the three days' pay... That would be equivalent to an overtime rate equal to double-time-and-a-half... counsel for the applicants submitted that 'earned lay-day pay' referred to pay to be earned for working on a lay-day, in addition

to regular pay for a lay-day already earned by previous service... **Key words:** [Labour, Human Rights, Patent Unreasonable, Judicial Review].

- **Case 3:** ... On November 13, 2001, the applicants noticed that the master schedule recently posted included working shifts on both December 25, 2001 and January 1, 2002, which are DPHs [Designated Paid Holidays]. The applicants advised their supervisor that this was contrary to clause 30.06 of the Collective Agreement... As the applicants declined to make such an election, the management 'H'ed' [placed on paid holiday leave] one half of the applicants on Christmas day and the other half on New Year's Day... They also asked that the master schedule be amended such that everyone could have either Christmas or New Year's Day as a day of rest. Counsel for the grievors contends that the employer should have changed their scheduled shifts to days of rest. In their view, the regime established by the Collective Agreement and the VSSA taken together requires that all days including December 25 and January 1 be included in the 56-day schedule either as work days or days of rest... **Key words:** [Labour, Tax, Administrative Law, Administrative, Judicial Review, Costs].

Failure of Baselines. Standard BM25 incorrectly ranks Case 3 higher than Case 2. This is due to significant lexical overlap in procedural keywords like "Judicial Review" and "Costs" shared between Case 1 and 3. Similarly, a direct LLM embedding yields a cosine similarity of 0.658 for the Positive (Case 2) versus 0.718 for the Negative (Case 3), failing to distinguish the nuanced legal issue.

Effectiveness of LSE. The LSE module precisely extracts key legal phrases like "Patent Unreasonable" (a standard of review standard). With LSE-enhanced summaries, the similarity gap narrows (Case 2: 0.692 vs. Case 3: 0.725), though the ranking remains incorrect.

Hierarchy and Graph Resolution. The HSE module generates the following discrete ID sequences ($L = 6$):

- **Case 1:** `[11, 48, 9, 37, 3, 6]`
- **Case 2:** `[11, 48, 9, 45, 14, 6]`
- **Case 3:** `[11, 48, 9, 37, 28, 11]`

All three cases share the same shallow ids `[11, 48, 9]`, correctly clustering them into the broad "Labor Law" domain. Interestingly, Case 3 shares the mid-level id `37` with Case 1, explaining why baselines find it similar (both involve Collective Agreement disputes). However, at the deepest layer (fine-grained residuals), Case 1 and Case 2 align on ID **6**, while Case 3 diverges to 11. This deep match captures the subtle factual nuance: both Case 1 and 2 specifically concern *monetary compensation calculations for work on rest days*, whereas Case 3 concerns *scheduling*.

Finally, the CRL module aggregates these hierarchical signals. By leveraging the shared deep node (ID 6) as a bridge, the graph amplifies the connection between Case 1 and 2. The final scores (Case 2: **0.784** > Case 3: 0.736) successfully reverse the ranking, demonstrating DSLaw's ability to prioritize fine-grained legal substance over surface-level lexical overlap.

D.4 Efficiency Analysis

To evaluate the efficiency of our approach, we compare retrieval time with representative baselines under identical hardware conditions. We report two metrics: Total Avg., which denotes the average total retrieval time, and Query Avg., which denotes the average time per query. For consistency, we use 400 queries for COLIEE2025 and 177 queries for ELAM. All experiments are conducted on a machine with two Intel Xeon Silver 4314 CPUs, 128 GB of RAM, and one NVIDIA A800 GPU.

The results are summarized in Tab. 7. All time measurements are reported in milliseconds with two decimal places, and are averaged by method type. For dense retrieval models, a preliminary filtering step is applied to select the top 30 candidates before reranking, whereas structure-aware models are evaluated directly under their standard retrieval settings.

In DSLaw, efficiency is mainly supported by compact representations and sparse graph computation. The LSE module reduces both token length and embedding dimensionality before downstream modeling. In CRL, GAT parameters are shared across graph layers to reduce model overhead. Graph construction is also sparse: each case node has only two explicit edges, linking it to its charge node and its layer-specific code-ID node, while charge–charge and ID–ID edges are maintained at bounded degrees. As a result, the total number of edges remains $O(n)$, where n is the number of nodes, which keeps graph propagation efficient. The total number of parameters in DSLaw

Category	COLIEE2025		ELAM	
	Total Avg.	Query Avg.	Total Avg.	Query Avg.
Traditional Retrieval Methods	412,848.50	1,032.12	37,107.50	209.64
LLM-based Methods	490,338.33	1,225.85	237,957.00	1,344.39
Structure-aware Models	112,096.00	280.24	96,418.00	544.73
DSLaw	107,580.30	268.95	66,337.50	374.79

Table 7: Average comparison of Total Avg. and Query Avg. across different categories of models on COLIEE2025 and ELAM datasets (batch size = 32). Results are reported with two decimal places.

Thereshold	R@1	R@5	R@10	N@1	N@5	N@10	P@1	P@5	P@10	MRR	MAP
COLIEE											
PCA-90%	0.0274	0.1072	0.1596	0.0976	0.1017	0.1188	0.0972	0.0793	0.0672	0.2043	0.0978
PCA-95%	0.0296	0.1110	0.1730	0.1017	0.1085	0.1299	0.1073	0.0854	0.0710	0.2201	0.1012
PCA-97%	0.0287	0.1116	0.1742	0.1007	0.1074	0.1316	0.1032	0.0848	0.0726	0.2204	0.1016
ELAM											
PCA-90%	0.0274	0.1072	0.1596	0.0976	0.1017	0.1188	0.0972	0.0793	0.0672	0.2043	0.0978
PCA-95%	0.0296	0.1110	0.1730	0.1017	0.1085	0.1299	0.1073	0.0854	0.0710	0.2201	0.1012
PCA-97%	0.0287	0.1116	0.1742	0.1007	0.1074	0.1316	0.1032	0.0848	0.0726	0.2204	0.1016

Table 8: Sensitivity analysis of PCA-whitening variance retention ratio on two datasets.

is approximately 734K. Moreover, because each graph layer uses a 2-layer GAT, inference can be carried out on a query-centered 2-hop subgraph rather than the full graph, which further reduces retrieval latency.

The LSE module also provides substantial compression before retrieval. On COLIEE2025, the average token count is reduced from 6244 to 1554 after salient sentence extraction, and further to 513 after LLM summarization. On ELAM, the corresponding numbers are 1485, 664, and 259. The average summary generation time is about 2.43 seconds per case on COLIEE2025 and 1.68 seconds per case on ELAM. In both datasets, the embedding dimensionality is reduced from 4096 to 512 after PCA-whitening.

D.5 Parameter Sensitivity Analyses

We conduct the following parameter sensitivity analyses:

PCA Variance Retention. We analyze the effect of PCA variance retention on both embedding dimensionality and retrieval performance. As shown in Tab. 8, retaining 97% of the variance results in only about 35% dimensionality reduction, whereas retaining 95% allows up to 75% compression. Although 97% yields slightly better performance on a few metrics, it substantially increases the representation size. In contrast, 95% provides a better balance between efficiency and performance, and is therefore adopted as the default setting.

Codebook Usage. As shown in Fig. 7 and Fig. 8, in both datasets, the codebook usage rate significantly decreases when the number of codebook layers is less than six, greatly limiting the expressive power of the codebook.

Layers in HSE. We further analyze the effect of the number of codebook layers in HSE. Fig. 9 and Fig. 10 show that using fewer codebook layers leads to worse retrieval performance on both datasets. This suggests that a shallow hierarchy is insufficient to capture the coarse-to-fine semantic distinctions required by the downstream graph reasoning process. Correspondingly, the adopted 6-layer setting provides a better trade-off between representational granularity and retrieval effectiveness.

Layers in CRL. We also evaluate the effect of the number of GNN layers in CRL. We test four settings with layer counts in $\{1, 2, 3, 4\}$, while fixing both input and output dimensions to 512. For multi-layer settings, the hidden dimension is set to 16 with 4 attention heads.

As shown in Fig. 11 and Fig. 12, a single-layer GNN performs worse than deeper configurations, while a 2-layer GNN achieves the best overall results on both datasets. Increasing the depth beyond two layers leads to slight performance degradation, suggesting that shallow graph propagation is sufficient to capture the most useful cross-case interactions in this task.

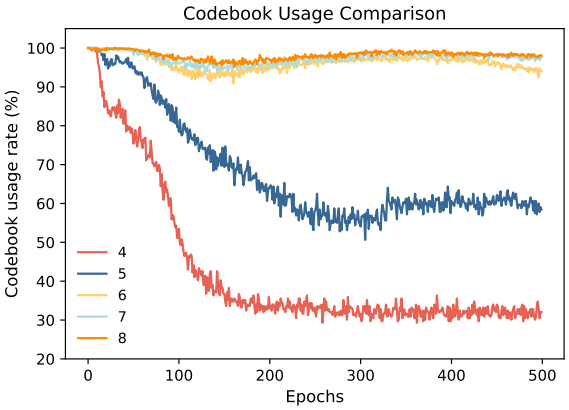

Figure 7: The codebook usage rate on dataset COLIEE2025 for different codebook layer configurations, with layers from 4 to 8.

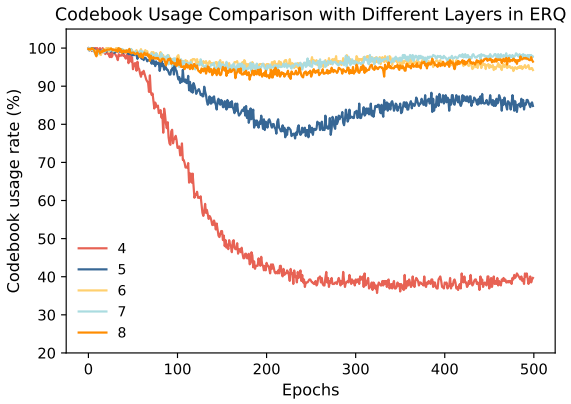

Figure 8: The codebook usage rate on dataset ELAM for different codebook layer configurations, with layers from 4 to 8.

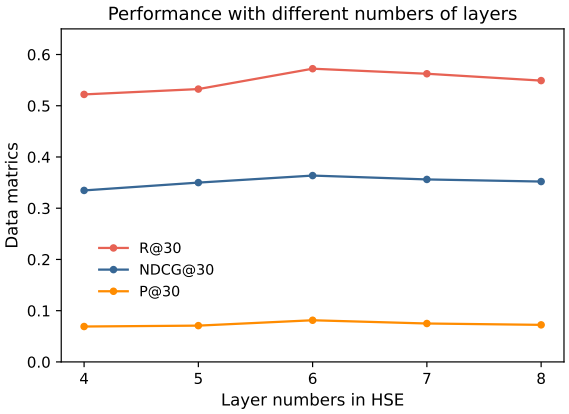

Figure 9: Comparison of multiple indicator data across different layer configurations of the HSE module on COLIEE2025.

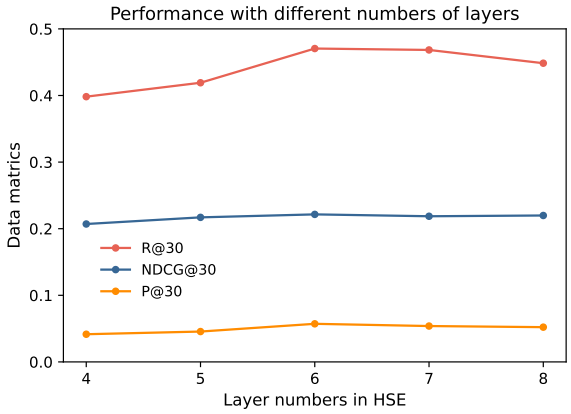

Figure 10: Comparison of multiple indicator data across different layer configurations of the HSE module on ELAM.

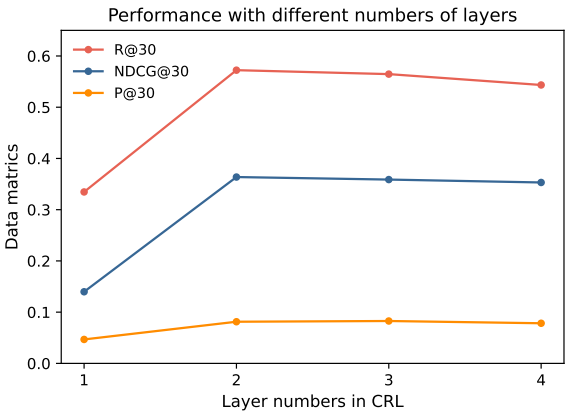

Figure 11: Comparison of multiple indicator data across different layer configurations of the CRL module on COLIEE2025.

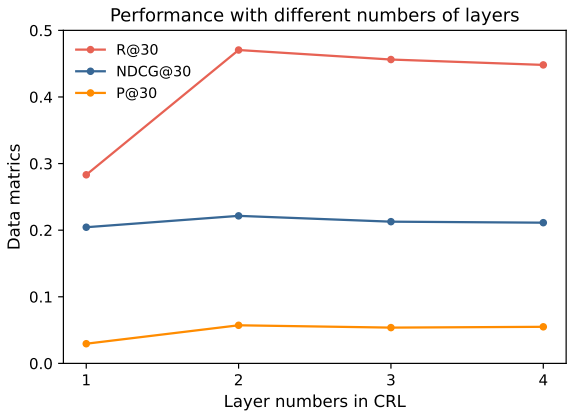

Figure 12: Comparison of multiple indicator data across different layer configurations of the CRL module on ELAM.

Charge	R@5	R@10	N@5	N@10	P@5	P@10
I	0.130	0.192	0.110	0.139	0.087	0.073
II	0.092	0.166	0.095	0.113	0.077	0.064
III	0.137	0.187	0.105	0.128	0.092	0.076
IV	0.117	0.157	0.098	0.118	0.086	0.067

Table 9: Stratified retrieval performance of DSLaw across four representative charge categories in the ELAM dataset.

D.6 Stratified Semantic Analysis across Charge Types

To further investigate how the semantic properties of the case corpora affect retrieval performance, and to provide a basis for deeper semantic explanation, we conduct a stratified analysis across different legal charge categories in the ELAM dataset. Specifically, we report the retrieval performance of DSLaw separately for four representative charge categories:(I) Crime of Obstructing Official Business; (II) Crime of Using Fake or Stolen Identity Documents;(III) Crime of Sabotaging Computer Information Systems; (IV) Crime of Organizing Cheating in Exams.

The results, summarized in Tab. 9, indicate that DSLaw maintains stable performance across diverse charge types. Although the absolute values naturally fluctuate due to varying data distributions and intrinsic case complexities, the retrieval metrics remain highly comparable across most categories.

Notably, we observe that Charge II exhibits relatively lower metrics compared to the other three categories. A plausible explanation is that this specific charge covers multiple closely related sub-types (e.g., forging, altering, or trading identity documents). These sub-types often share highly similar factual patterns and vocabularies, yet differ in their specific legal qualifications. Such intra-category semantic proximity increases the difficulty of fine-grained distinction, which partially explains the comparatively lower scores.

Nevertheless, DSLaw maintains robust and stable behavior even under this more challenging setting. This stratified assessment demonstrates that our hierarchical discrete representation remains effective at capturing subtle legal boundaries, providing a clear picture of where the model excels and where extreme intra-category proximity introduces challenges.